UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/687,233	10/15/2003	Martin Runte	6570P015	8107

45062 7590 06/06/2008

SAP/BLAKELY
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
06/06/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/687,233	RUNTE ET AL.	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 March 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-6,8,11-19,21,22,24,25,32,34 and 35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,4-6,8,11-19,21,22,24,25,32,34 and 35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the RCE filed on March 20, 2008.

2. **Claims 1, 4-6, 8, 11-19, 21, 22, 24, 25, 32, 34, and 35** are pending.

3. **Claims 1, 4, 6, 8, 12, 22, and 32** have been amended.

4. **Claims 2, 3, 7, 9, 10, 20, 23, 26-31, 33, and 36-38** have been cancelled.

5. The objections to Claims 26-31 are withdrawn in view of Applicant's cancellation of the claims.

6. The 35 U.S.C. § 112, first paragraph, rejections of Claims 26-31 are withdrawn in view of Applicant's cancellation of the claims.

7. The 35 U.S.C. § 101 rejections of Claims 26-31 are withdrawn in view of Applicant's cancellation of the claims.

Response to Amendment

Claim Objections

8. **Claims 1, 4-6, 8, 11-19, 21, 22, 24, 25, 32, 34, and 35** are objected to because of the following informalities:

- **Claims 1, 6, and 32** recite the limitation "the second subsystem." Applicant is advised to change this limitation to read "the second software subsystem" for the purpose of providing it with proper explicit antecedent basis.
- **Claims 4 and 5** depend on Claim 1 and, therefore, suffer the same deficiency as Claim 1.

- **Claims 8, 11-19, and 21** depend on Claim 6 and, therefore, suffer the same deficiency as Claim 6.
- **Claims 34 and 35** depend on Claim 32 and, therefore, suffer the same deficiency as Claim 32.
- **Claims 4, 5, 8, 11-19, 21, 24, and 25** recite the category of invention "[t]he method." Applicant is advised to change this category of invention to read "[t]he computer-implemented method" for the purpose of providing it with proper explicit antecedent basis.
- **Claim 21** contains a typographical error: Claim 21 should presumably depend on Claim 6, not Claim 7.
- **Claim 22** contains the following typographical errors:
  - "monitoring a software development space to detect changes *in* introduced into a subset of frozen software objects ..." should read -- monitoring a software development space to detect changes introduced into a subset of frozen software objects ... --.
  - "determining whether any changes were introduced into *a* the subset of frozen software objects" should read -- determining whether any changes were introduced into the subset of frozen software objects --.
- **Claim 22** recites the limitations "a first software system" and "a/the frozen object" Applicant is advised to change these limitations to read "a first software subsystem" and "a/the frozen software object," respectively, for the purpose of keeping the claim language consistent throughout the claims.

- **Claims 24 and 25** depend on Claim 22 and, therefore, suffer the same deficiencies as Claim 22.

Appropriate correction is required.

9. **Claim 24** is objected to under 37 CFR 1.75(c), as being of improper dependent form for failing to further limit the subject matter of a previous claim. Applicant is required to cancel the claim(s), or amend the claim(s) to place the claim(s) in proper dependent form, or rewrite the claim(s) in independent form.

Claim 24 recites the limitation "wherein the subset of the software objects includes frozen software objects." The limitation does not constitute a further limitation of Claim 22 because Claim 22 already recites the limitation "a subset of frozen software objects."

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. **Claims 1, 4, 32, and 34** are rejected under 35 U.S.C. 102(b) as being anticipated by **US 2002/0072928 (hereinafter "Sundararajan")**.

As per **Claim 1**, Sundararajan discloses:

- automatically detecting a change introduced into a software object of a first software subsystem, where the first software subsystem stores the software object from which instances of the software object are created, wherein the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem *(see Figure 1; Paragraph [0016], "An exemplary relationship within an e-commerce application might include, but is not limited to, identified compatibility between a web application server and web server software, which both cooperate to present one or more web pages to an end user, such as a user of a user computer."; Paragraph [0029], "In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features (the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem)."; Paragraph [0041], "The combination of hardware and software represented on each e-commerce computer 14a, 14b and 14c is hereinafter collectively referred to as "components.""; Paragraph [0045], "At step 150, an operator of any one of the e-commerce servers 14a, 14b and/or 14c can elect to revise a current application configuration definition associated with the business models or production environments, which are hosted on any one of the e-commerce servers 14a, 14b and/or 14c (where the first software subsystem stores the software object from which instances of the software object are created). At step 160, the operator can execute the revision of the current application configuration definition, which is associated with the business models or production*

environment, by installing a new component or revising an existing component (automatically detecting a change introduced into a software object of a first software subsystem.");

- accessing a compatibility changes database in response to detecting the change, where the compatible changes database indicates changes defined to be compatible with the software objects of the second software subsystem *(see Paragraph [0038], "The component change manager 12 further includes a database 25 having a plurality of records A, B and C defined therein. More specifically, the records A, B and C each include a plurality of data fields and associated control fields related to inter-relationships of components located on each of the e-commerce servers 14a, 14b and 14c."; Paragraph [0046], "After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships.")*;

- determining, based on accessing the compatible changes database, whether the change is compatible with the software objects of the second software subsystem, including determining whether the change is predefined as compatible *(see Paragraph [0046], "If the component revision is verified as compatible, at step 180, the component revision is executed at step 190.")*; and

- implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem *(see Paragraph [0046], "If the component revision is verified as compatible, at step 180, the*

component revision is executed at step 190. This component revision becomes the current application configuration definition for the production environment hosted on the e-commerce server 14a, 14b and/or 14c."); otherwise,

- rejecting the introduced change and generating an error notification *(see Paragraph [0022], "Configuration maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration."; Paragraph [0046], "If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.")*.

As per **Claim 4**, the rejection of **Claim 1** is incorporated; and Sundararajan further discloses:

- issuing a message that the change is not allowed if the change is not predefined as compatible *(see Paragraph [0022], "Configuration maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration."; Paragraph [0046], "If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.")*.

Claims 32 and 34 are article of manufacture claims corresponding to the computer-implemented method claims above (Claims 1 and 4) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1 and 4.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. **Claims 5 and 35** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **US 6,658,659 (hereinafter "Hiller")**.

As per **Claim 5**, the rejection of **Claim 4** is incorporated; however, Sundararajan does not disclose:

- allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

Hiller discloses:

- allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible *(see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility*

according to a minimum version number or a version number corresponding to a minimum date of release.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller into the teaching of Sundararajan to include allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released *(see Hiller – Column 13: 6-12).*

Claim 35 is rejected for the same reason set forth in the rejection of Claim 5.

14. **Claims 6, 13-15, 17-19, 22, 24, and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **US 6,725,452 (hereinafter "Te'eni")**.

As per **Claim 6**, Sundararajan discloses:

- identifying a subset of software objects of a first software subsystem that are stored in the first software subsystem and are to be instantiated in a second software subsystem to interact with software objects of the second software subsystem *(see Figure 1; Paragraph [0016], "An exemplary relationship within an e-commerce application might include, but is not limited to, identified compatibility between a web application server and web server software, which both cooperate to present one or more web pages to an end user, such as a user of a user computer.";*

Paragraph [0029], "In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features (instantiated in a second software subsystem to interact with software objects of the second software subsystem)."; Paragraph [0041], "The combination of hardware and software represented on each e-commerce computer 14a, 14b and 14c is hereinafter collectively referred to as "components.""; Paragraph [0045], "At step 150, an operator of any one of the e-commerce servers 14a, 14b and/or 14c can elect to revise a current application configuration definition associated with the business models or production environments, which are hosted on any one of the e-commerce servers 14a, 14b and/or 14c (identifying a subset of software objects of a first software subsystem that are stored in the first software subsystem).);

- detecting a change introduced into a software object from the subset of software objects; and prior to allowing the change *(see Paragraph [0045], "At step 160, the operator can execute the revision of the current application configuration definition, which is associated with the business models or production environment, by installing a new component or revising an existing component.")*,

- accessing a compatible changes database in response to detecting the change, where the compatible changes database indicates changes predefined to be compatible with the software objects of the second software subsystem *(see Paragraph [0038], "The component change manager 12 further includes a database 25 having a plurality of records A, B and C defined therein. More specifically, the records A, B and C each include a plurality of data fields and associated control fields related to inter-relationships of components located on each of the e-*

commerce servers 14a, 14b and 14c."; Paragraph [0046], "After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships.");

- determining with a compatibility check, based on accessing the compatible changes database, whether the change is compatible with a second software subsystem, including determining whether the change is predefined as compatible *(see Paragraph [0046], "If the component revision is verified as compatible, at step 180, the component revision is executed at step 190.")*; and

- issuing a notice indicating results of the compatibility check *(see Paragraph [0022], "Configuration maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration."; Paragraph [0046], "If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.").*

However, Sundararajan does not disclose:

- declaring the subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined to be compatible or the changes are approved by an expert in compatibility between the software subsystems.

Te'eni discloses:

- declaring the subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined to be compatible or the changes are approved by an expert in compatibility between the software subsystems *(see Column 14: 32 and 33, "LOCKS a set of components that were locked by the user in order to present installation or removal thereof ..." and 46-53, "At step 90, LOCKS is examined to check whether LOCKS contains B.CI. If the result of step 90 is positive, at step 92 MODE is examined to check whether the value thereof is "interactive mode". If the result is negative, RESULT is returned to the calling routine with the value of "failure". If the result of step 92 is positive, the user's permission is requested at step 96 to remove the lock from the component." and 55-57, "In contrast, if the user allows the removal of the lock from the component at step 100, a series of tests are made at steps 98, 104, and 106.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Te'eni into the teaching of Sundararajan to include declaring the subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined to be compatible or the changes are approved by an expert in compatibility between the software subsystems. The modification would be obvious because one of ordinary skill in the art would be motivated to resolve inter-component compatibilities among hardware and software entities *(see Te'eni – Column 15: 9-14).*

As per **Claim 13**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object is a function module *(see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters.").*

As per **Claim 14**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object is a data structure *(see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters.").*

As per **Claim 15**, the rejection of **Claim 13** is incorporated; and Sundararajan further discloses:

- wherein the software object includes an environment of the function module *(see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters.").*

As per **Claim 17**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object includes an interface and an environment of the interface *(see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters.").*

As per **Claim 18**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object includes a program and an environment of the program *(see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters.").*

As per **Claim 19**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein the detecting the change comprises automatically monitoring development of software code *(see Paragraph [0045], "At step 160, the operator can execute the revision of the current application configuration definition, which is associated with the business models or production environment, by installing a new component or revising an existing component.").*

As per **Claim 22**, Sundararajan discloses:

- monitoring a software development space to detect changes introduced into a subset of software objects stored in a first software subsystem that are to be instantiated in a second software subsystem *(see Figure 1; Paragraph [0016], "An exemplary relationship within an e-commerce application might include, but is not limited to, identified compatibility between a web application server and web server software, which both cooperate to present one or more web pages to an end user, such as a user of a user computer."; Paragraph [0029], "In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple*

configurations over a range of version levels with a variety of enabled features."; Paragraph [0041], "The combination of hardware and software represented on each e-commerce computer 14a, 14b and 14c is hereinafter collectively referred to as "components.""; Paragraph [0045], "At step 150, an operator of any one of the e-commerce servers 14a, 14b and/or 14c can elect to revise a current application configuration definition associated with the business models or production environments, which are hosted on any one of the e-commerce servers 14a, 14b and/or 14c. At step 160, the operator can execute the revision of the current application configuration definition, which is associated with the business models or production environment, by installing a new component or revising an existing component.");

- performing a global compatibility check of the subset of software objects of the first software subsystem by determining whether any changes were introduced into the subset of software objects of the first software subsystem since the time of a last compatibility check wherein the introduced changes were unapproved changes introduced without obtaining prior approval, where performing the global compatibility check includes comparing a current version of software code of a software object in the software development space with a version of the software code of the software object at the time of a last global compatibility check *(see Paragraph [0022], "A representative corrective action may include restoring a specified version of a component to a previous condition or status upon system detection of an error in a pre-determined relationship (wherein the introduced changes were unapproved changes introduced without obtaining prior approval)."; Paragraph [0023], "Verifying and maintaining the integrity of the application configuration definition may occur at random intervals or at predetermined intervals." and "Revising existing components and/or adding new components,*

such as adding enhancements to specific components of a web application server, can expose a stable executing configuration to a potential operating error if the relationship of the components are not verified. Thus, verifying and maintaining the integrity of the application configuration definition is necessary before and after upgrading the e-commerce application with changes to the associated web application server software (performing a global compatibility check of the subset of software objects of the first software subsystem by determining whether any changes were introduced into the subset of software objects of the first software subsystem since the time of a last compatibility check). Subsequent verification and maintenance checkpoints may be required depending on the significance of the relationship to be validated and impact to the system if a relationship becomes corrupted between checkpoints."; Paragraph [0046], "After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships (comparing a current version of software code of a software object in the software development space with a version of the software code of the software object at the time of a last global compatibility check).");

- identifying software objects of the second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software system *(see Paragraph [0046], "After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce*

server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships." and "If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt."); and

- issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change *(see Paragraph [0022], "Configuration maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration."; Paragraph [0046], "If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.").*

However, Sundararajan does not disclose:

- a subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined as compatible or the changes are approved by an expert in compatibility between the software subsystems.

Te'eni discloses:

- a subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined as compatible or the changes are approved by an expert in compatibility between the software subsystems *(see Column 14: 32 and 33, "LOCKS a set of components that were locked by the user in order to present installation or removal thereof ..." and 46-53, "At step 90, LOCKS is examined to check whether LOCKS contains B.CI. If the*

result of step 90 is positive, at step 92 MODE is examined to check whether the value thereof is "interactive mode". If the result is negative, RESULT is returned to the calling routine with the value of "failure". If the result of step 92 is positive, the user's permission is requested at step 96 to remove the lock from the component." and 55-57, "In contrast, if the user allows the removal of the lock from the component at step 100, a series of tests are made at steps 98, 104, and 106.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Te'eni into the teaching of Sundararajan to include a subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined as compatible or the changes are approved by an expert in compatibility between the software subsystems. The modification would be obvious because one of ordinary skill in the art would be motivated to resolve inter-component compatibilities among hardware and software entities *(see Te'eni – Column 15: 9-14).*

As per **Claim 24**, the rejection of **Claim 22** is incorporated; however, Sundararajan does not disclose:

- wherein the subset of the software objects includes frozen software objects.

Te'eni discloses:

- wherein the subset of the software objects includes frozen software objects *(see Column 14: 32 and 33, "LOCKS a set of components that were locked by the user in order to present installation or removal thereof ...").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Te'eni into the teaching of Sundararajan to include wherein the subset of the software objects includes frozen software objects. The modification would be obvious because one of ordinary skill in the art would be motivated to prevent inter-component compatibilities among hardware and software entities *(see Te'eni – Column 14: 32 and 33)*.

As per **Claim 25**, the rejection of **Claim 24** is incorporated; and Sundararajan further discloses:

- wherein the frozen software objects include software objects of the first software subsystem used by software objects of the second software subsystem *(see Paragraph [0029], "In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features.").*

15. **Claims 8, 11, and 12** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **Te'eni** as applied to Claims 6 above, and further in view of **US 2004/0230952 (hereinafter "Massaro")**.

As per **Claim 8**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein frozen software objects are classified to include released objects and restricted objects based on a number of instances instantiated in the second software subsystem *(see Paragraph [0042], "More specifically, the plurality of permutations of dynamic relationships or inter-relationships are determined by determining all possible combinations of components (released objects and restricted objects having a number of instances) that can be executed to carry out an e-commerce transaction as well as all possible combinations of component relationships and inter-relationships associated with each combination of components."; Paragraph [0043], "For example, one permutation of an e-commerce transaction can include a user at user computer 20a, 20b and/or 20c communicating an e-commerce transaction to the e-commerce server 14a, 14b, and/or 14c ...").*

However, Sundararajan and Te'eni do not disclose:

- where released objects are defined as software objects having a number of instances instantiated in the second software subsystem exceeds a threshold number, and restricted objects are defined as software objects having a number of instances instantiated in the second software subsystem that does not exceed the threshold number.

Massaro discloses:

- using a threshold number to distinguish between versions of files both when the number of changes to a region is few and when the number of changes is large *(see Paragraph [0010], "A method, apparatus, system, and signal-bearing medium are provided that in an embodiment compare versions and mark an entire region as changed when the number of changes to the region exceeds a threshold. In contrast, when the number of changes in a region does not exceed the threshold, the individual changes are marked. In this way, a user can*

distinguish between versions both when the number of changes to a region is few and when the number of changes is large.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Massaro into the teaching of Sundararajan to include where released objects are defined as software objects having a number of instances instantiated in the second software subsystem exceeds a threshold number, and restricted objects are defined as software objects having a number of instances instantiated in the second software subsystem that does not exceed the threshold number. The modification would be obvious because one of ordinary skill in the art would be motivated to distinguish between components based on a usage limit.

As per **Claim 11**, the rejection of **Claim 8** is incorporated; and Sundararajan further discloses:

- wherein an identification of recent changes introduced into a restricted object is provided when software objects of the second software subsystem request new usage of the restricted object *(see Paragraph [0044], "The static and dynamic inter-relationships, including the plurality of permutations of dynamic inter-relationships, can be associated with data fields. The data fields are encoded control fields identifying the compatibility of one component (e.g., web server software) with another component (e.g., web application software). The data fields are stored in a record A, B, or C defined on the database 25, which database 25 is represented on the component change manager 12.").*

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Sundararajan further discloses:

- wherein classification of each frozen software object is based on a number of instances of each frozen software object occurring in the second software subsystem *(see Paragraph [0042], "More specifically, the plurality of permutations of dynamic relationships or inter-relationships are determined by determining all possible combinations of components that can be executed to carry out an e-commerce transaction as well as all possible combinations of component relationships and inter-relationships associated with each combination of components."; Paragraph [0043], "For example, one permutation of an e-commerce transaction can include a user at user computer 20a, 20b and/or 20c communicating an e-commerce transaction to the e-commerce server 14a, 14b, and/or 14c ...").*

16. **Claim 16** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **Te'eni** as applied to Claim 6 above, and further in view of **US 6,415,435 (hereinafter "McIntyre")**.

As per **Claim 16**, the rejection of **Claim 6** is incorporated; however, Sundararajan and Te'eni do not disclose:

- wherein a software object includes a class and an environment of the class.

McIntyre discloses:

- wherein a software object includes a class and an environment of the class *(see Column 1: 8-14, "The present invention relates generally to an improved data processing system*

and in particular to a method and apparatus for determining compatibility of different classes in a data processing system.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of McIntyre into the teaching of Sundararajan to include wherein a software object includes a class and an environment of the class. The modification would be obvious because one of ordinary skill in the art would be motivated to determine compatibility of different classes *(see McIntyre – Column 1: 8-14).*

17. **Claim 21** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **Te'eni** as applied to Claim 6 above, and further in view of **Hiller**.

As per **Claim 21**, the rejection of **Claim 6** is incorporated; however, Sundararajan and Te'eni do not disclose:

- wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible.

Hiller discloses:

- wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible *(see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller into the teaching of Sundararajan to

include wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released *(see Hiller – Column 13: 6-12)*.

Response to Arguments

18. Applicant's arguments with respect to Claims 1, 6, 22, and 32 have been considered, but are moot in view of the new ground(s) of rejection.

Conclusion

19. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

20. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/QC/
May 21, 2008

/Wei Zhen/

Supervisory Patent Examiner, Art Unit 2191